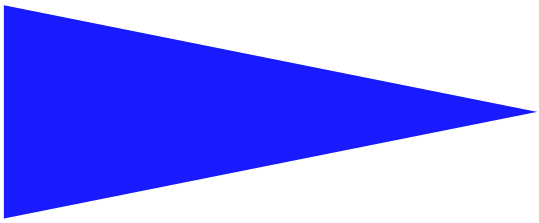
Deterministic delta-Connected Overlay for Peer-to-Peer Networks

Ajoy Datta, Maria Gradinariu, Antonino Virgillito

I R I S A
INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTÈMES ALÉATOIRES

PUBLICATION
INTERNE
N° 1739



DETERMINISTIC δ-CONNECTED OVERLAY FOR PEER-TO-PEER NETWORKS

A. K. DATTA M. GRADINARIU A. VIRGILLITO

ISSN 1166-8687

IRISA
CAMPUS UNIVERSITAIRE DE BEAULIEU - 35042 RENNES CEDEX - FRANCE

IRISA

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTÈMES ALÉATOIRES
Campus de Beaulieu – 35042 Rennes Cedex – France
Tél. : (33) 02 99 84 71 00 – Fax : (33) 02 99 84 71 71
http://www.irisa.fr

Deterministic δ-Connected Overlay for Peer-to-Peer Networks

A. K. Datta* M. Gradinariu A. Virgillito **

Systèmes communicants
Projet Adept

Publication interne n ° 1739 — Mars 2006 — 10 pages

Abstract: The network connectivity is a basic requirement while implementing fundamental communication and storage abstractions in P2P networks, featuring scalability and fault-tolerance. The quality of services of abstractions like for example multicast, publish/subscribe, group membership or persistent storage is strongly related to the connectivity degree of the underlying overlay. Intuitively, a higher overlay connectivity ensures a reinforced reliability and consequently, the deployment of distributed applications with real-time constraints on top of these overlays becomes feasible even in environments characterized by a high dynamicity, i.e., nodes arriving and departing at a high rate.

Our paper proposes a novel δ-connected DHT-free P2P overlay. Our overlay offers strong connectivity guarantees despite the system dynamicity. The construction and the maintenance of our overlay is completely decentralized and handled strictly locally, through deterministic algorithms which correctness is rigorously proved.

Key-words: connectivity, P2P overlays, fault tolerance, self-organization

(Résumé : tsvp)

* School of Computer Science, University of Nevada, Las Vegas (USA
** Università di Roma, “La Sapienza”, Italy

Infrastructure deterministe δ-Connectée pour les rseaux de pairs

Résumé : La connectivitée d'un réseaux est une des briques essentielle à la mise en place de services tolérants aux fautes dans les systèmes grande échelle. Nous proposons une infrastructure pair-à-pair déterministe et δ-connectée. La mise en place de notre infrastructure est totalement décentralisée et utilise uniquement des informations locales.

Mots clés : connectivité, P2P overlays, tolerance aux fautes, auto-organisation

1 Introduction

Peer-to-peer systems are a popular paradigm for distributed systems, mainly thanks to the widespread use of file-sharing applications. Recently they inspired a massive amount of research targeted at identifying a broader range of applications (such as persistent storage or multicast and publish/subscribe diffusion), that can exploit their interesting characteristics.

In a peer-to-peer system, participants self-organize in a completely decentralized fashion and have only a partial knowledge of other participants. Thanks to this, such systems are able to scale to massive proportions. A basic problem to be solved in this context is to guarantee that the overlay obtained considering the mutual knowledge among participants is still connected despite frequent changes in the composition of the system. A partitioned graph would present communities of isolated participants, that does not exploit and at the same time degrade the application functionality. An algorithm for the construction of a peer-to-peer overlay arranges the connections among nodes as they join the system and maintains the connectivity when nodes in the system gracefully leave or abruptly fail. DHTs (Distributed Hash Tables) were till recently considered an appealing tool for constructing efficient overlays. In a DHT-based overlay the network topology and the data placement are designed in order to provide upper bounds on the data search and usually, the complexity of a lookup operation in these systems is logarithmic with the number of nodes in the system. The drawbacks of DHT-based overlays are their high complexity, low robustness and adaptability in highly dynamic systems, and limited search capabilities (e.g. for keyword matching carries). For all these reasons the current research considers the DHT-free overlays. The challenge is to design overlays approximating the features of the DHT-based overlays without preserving the DHT's drawbacks.

Most DHT-free algorithms for overlay construction are based on gossiping, i.e. periodical exchange of neighborhood information between each node and a randomly chosen set of its neighbors. Such algorithms are typically evaluated through simulation or via probabilistic analysis, showing that the probability of partitions is very low as the network grows. The probabilistic methodology applies well to gossip-based algorithms, inherently characterized by random behavior. However, the existing probabilistic analysis for the most popular gossip-based systems do not give any insight concerning the behavior of the algorithm in situations characterized by a high churn (i.e. nodes frequently and continuously connecting and disconnecting to/from the system). As shown in several studies [5], peer-to-peer systems are commonly subject to high churn rates. Recently, Baldoni et al. [4] showed that one of the most popular membership algorithms, namely Scamp [9], though proved correct probabilistically, is prone to partitioning when subject to high churn rates. Moreover, the gossip-based methods induce a high load, as many duplicated messages are sent [8].

In this paper we propose a novel deterministic algorithm which constructs a δ-connected overlay through deterministic algorithms for construction and maintenance. Our overlay has three main features: (1) its diameter grows logarithmically with the number of nodes in the network; (2) it remains connected after random failures of a linear subset of its nodes and/or edges and (3) at least δ nodes need to be removed in order to cause a partition. Moreover, it works with limited knowledge of the system nodes. This knowledge is independent of the system size and does not use "privileged" nodes (i.e., nodes with a special function). We formally prove the correctness of our algorithm by showing that the algorithm is able to preserve and restore the connectivity of the network despite concurrent joins, leave and failures. Differently from other analysis of churn in peer-to-peer networks, we consider an asynchronous system model, with no bounds assumed on message delay and no synchronization of nodes required. Only an assumption is necessary, limiting the number of concurrent joins and leaves of connected nodes to δ: this means that the overlay can simply adapt to higher degree of concurrency simply by properly tuning δ.

The paper is organized as follows. Section 2 presents the related work. In Section 3 we give basic definitions and define the system model. Section 4 describes the membership algorithm in detail. Formal proofs of correctness are given in Section 5. Section 6 concludes the paper.

2 Related Work

Recently, several peer-to-peer deterministic and probabilistic overlay infrastructures were proposed for supporting fundamental services as for example the group membership, multicast or publish-subscribe.

The common goals of all these overlays was to ensure (a) the local handling of the join/leave operations; (b) the high reliability in the presence of message loss and nodes joins and leaves; (c) load balancing and (d) an efficient support for optimal communication.

The overlays which achieved almost all these desiderata are based on gossip [9, 20, 2, 1]. The gossip based techniques have been already proved undisrupted with high probability in the presence of message loss and nodes leave and join. However, the main drawback of these systems is the explosive number of messages used in the maintenance process.

Tree-based overlays are another interesting class of overlays [6, 12]. These overlays aim at ensuring an efficient communication medium, but they offer poor load balancing since inner nodes in the tree carry the most of the system load. Moreover, their reliability is poor in the presence of message loss and node joins/leaves. In order to correct the latter problem some systems use long links, i.e. additional connections that enhance the overlay resilience to dynamicity. On the contrary, no viable solution was proposed so far for the load balancing aspect.

Another class of overlays [18, 19, 17, 16, 15] uses DHT techniques and aims at reducing the complexity of lookup operations and balance the system load. Thus, they have become an eligible support for efficient communication. However, the scalability and the reliability of these systems is limited, being often subject to partitions when facing a high churn. Moreover, the complexity of join operations in these systems is logarithmic in the network size since nodes may search and locate their position *a priori* established by the DHT key allocation algorithm.

Recently, several DHT-based overlays addressed the reliability problem ([13, 10, 11]). However, none of these systems lowered the logarithmic complexity of join operations. Moreover, in [7] the authors advocate that these systems are not scalable since they do not offer undisrupted service in the presence of high churn.

Very recently, in [14] the authors proposed a deterministic DHT-free overlay (Araneola) approximating the properties of a k-regular random graph, specifically (i) diameter growing logarithmic with network size (ii) connectivity despite random failures of a linear subset of nodes. However, Araneola only asymptotically converges to the above mentioned features when no join and leave operations stop to be executed. No analysis is proposed for the case when joins and leaves execute in parallel with the overlay maintenance. In this paper we propose a deterministic δ-connected overlay which approximates the k-regular graphs features despite frequent joins and leaves. We also identify the necessary and sufficient conditions related to the dynamic operations in order to maintain the δ-connectivity of overlay. Note that the main difference between our work and [14] resides in achieving the reinforced overlay connectivity. In [14] this property is achieved only eventually while in our system it is always guaranteed.

3 System Model

We consider a distributed dynamic system composed of a finite yet unbounded set of processes. The network is described by a weakly connected graph. Its nodes represent processes of the system and its edges represent established communication links between processes. The graph is referred in the following as the communication graph.

In order to connect to the system, a process executes an underlying connection (join) protocol. A process p is called *active* if there exists at least one process q which is already active in the system and aware of p. The set of *neighbors* of a process p is the set of processes q such that the link (p, q) is up (p and q are aware of each other and can communicate to each other through an underlying communication protocol). The logical graph defined by the neighbor relation is referred in the following as *overlay*. We assume the system to be subject to frequent and unpredictable changes: processes can join it or leave from it arbitrarily often, and they can fail temporarily (transient faults) or permanently (crash failures). Communication links can commit transient failures (e.g. messages loss).

To describe and analyze such distributed and dynamic system rigorously, we use the dynamic I/O automata introduced by Attie and Lynch [3]. This model allows the modeling of individual components, their interactions and their changes. The external actions of a dynamic I/O automata are classified in two categories, namely the input-output actions (I/O actions), and dynamic actions (C/D actions for Connection(Join)-Disconnection(Leave) actions) describing the mobility within the system. We introduce

the notion of configuration which is defined as the system state at time t altogether with the communication graph. An execution $e = (c_0, \ldots, c_t, \ldots)$ of a dynamic I/O automaton is an infinite sequence of configurations, where c_0 is the initial configuration of the system. Every transition $c_i \rightarrow c_{i+1}$ is associated to the execution of at least one of the previously defined actions.

In a trasition, two actions a_1 and a_2 are *locally concurrent* if they are concurrently executed by neighboring nodes. We impose a restriction on the local concurrency of operations: being δ a parameter of the system then in each neighbourhood no more than $\delta - 1$ locally concurrent join/leave actions can be executed during a transition.

4 A Protocol for a δ-connected Overlay Construction

In this section, we describe a protocol for constructing and maintaining a δ-connected overlay. The protocol requires bounded knowledge at nodes. The size of the partial view of each node is independent of the system size. The protocol is activated when a node u issues a join request at another node in the system (Join phase) or when nodes fail or voluntarily leave (Maintenance phase).

Each node in the system maintains bidirectional links to k other nodes (k is an application parameter). The objective of the join algorithm is to preserve the δ connectivity of the overlay where $\delta = \lfloor \frac{k}{2} \rfloor + 1$. Thus, up to $\delta - 1$ failures/leaves can be tolerated before the overlay partitions. The maintenance phase (repair operation) is invoked at each node and it periodically checks the node neighborhood. If some of its neighbors are failed then the maintenance phase locally repairs the overlay by replacing the dead links.

4.1 Data Structures

Each node maintains the following data structures:

- $neighbors$: list of k pointers to neighbors in the overlay. A $select(l, view)$ method is used to pick following an arbitrary strategy l nodes from the set $view$.
- $is\_saturated$: boolean variable indicating if the neighbors' view of the node is saturated, i.e., the number of known neighbors is equal to k.

4.2 Protocol Details

---

upon receive JOIN(new_node) at node x from node y
  $NS \leftarrow \{u \in neighbors : u.is\_saturated = false\}$;
  $S \leftarrow neighbors - NS$;
  if $\lfloor \frac{|S|}{2} \rfloor + |NS| \leq \lfloor \frac{k}{2} \rfloor$ **then** send JOIN(new_node) to select(1, $neighbors$)
  else
    $neighbors \leftarrow neighbors \cup new\_node$;
    $X \leftarrow$ select($\lfloor \frac{|S|}{2} \rfloor, S$);
    send PUSH_UPDATE($NS \cup X$) to new_node;
    send PUSH_UPDATE(new_node) to $NS \cup X$;
    $neighbors \leftarrow neighbors - X$;
    if $(is\_saturated) \wedge (X \neq \emptyset)$ **then** send SATURATED(false) to $neighbors$;
  end if
LEAVE() at node x
send(PUSH_UPDATE($neighbors$) to $neighbors$;

Figure 1: Pseudo-code of Join and Leave Operations

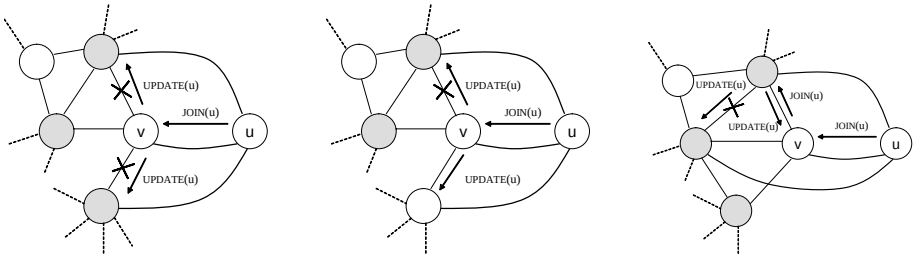


Figure 2: Node u joining to node v ($k = 5$). New links are depicted with thick lines. Note that the situation in (a) is not allowed by our protocol.

Join operation. A join operation is requested by a node u to a node v already in the system. v can be any node in the system with no particular properties and its address can be obtained for example from a set of dedicated servers. The pseudo-code of the JOIN message handler is presented in Figures 1.Adding u in the system requires that after the addition of u, the general connectivity of the overlay remains stable. More precisely, our algorithm guarantees that the overlay is at any time $\delta-$connected, with $\delta = \lfloor \frac{k}{2} \rfloor + 1$. v could provide u with the full list of its neighbors for it to connect to. However, the view of each node has limited size. Then, if one of v's neighbors has already a saturated view (i.e., the number of known neighbors is equal to k), it has to cut one of its existing links to create the new link with u (Figure 2(a)).

If all v's neighbors cut the link with v, the connectivity of the group may drop below the requirement (δ)[1].

Then, the join operation proceeds as follows: v first checks the sets S and NS of its saturated and non-saturated neighbors, respectively. This information is stored in a local variable whose value is updated when receiving a SATURATED message, sent by nodes themselves when they enter or exit a saturated state. v provides u with all its non-saturated neighbors plus $\lfloor \frac{|S|}{2} \rfloor$ among the saturated ones (Figure 2(b)). However, this is done only if $\lfloor \frac{|S|}{2} \rfloor + |NS| > \lfloor \frac{k}{2} \rfloor$. In this situation, the saturated nodes provided to u will cut their link with v but both v and u are guaranteed to have at least $\lfloor \frac{k}{2} \rfloor$ nodes after the join. If the condition is not satisfied, then v has to forward the join request to one of its neighbors (Figure 2(c)) [2].

View updates are invoked through a PUSH_UPDATE message containing as parameters the nodes that have to be added. Upon receiving a PUSH_UPDATE message a node adds the received nodes to its $neighbors$ list. If the node is saturated it removes from its view the node it received the update from.

Repair operation. Figure 3 shows the pseudo-code of the maintenance phase. Periodically each node contacts all the nodes in all its views for checking if they are still alive. If a node does not respond, it is immediately replaced through the PULL_NEIGHBORS request by another node avoiding that a sequence of failures generates a partitioning of the overlay. with some other nodes. Such nodes are requested by a node to some through PULL_NEIGHBORS. After a failure, a node can be pulled from other nodes while it is waiting for the reply of its own pull request. A node v does not reply to a PULL_NEIGHBORS request until it has at least δ elements in its $neighbors$ view. This is required for two reasons: i) providing the requester with nodes it does not already have and ii) avoiding that if the requester creates a connection with a saturated neighbor of v, this cuts the link with v, possibly lowering its degree below δ. We show in the correctness proof that nodes do not block indefinitely waiting for this condition to occur.

Finally, a node v voluntarily leaving the system, sends a PUSH_UPDATE with its neighbors list to all its neighbors. v's neighbors that did not know each other before v departure will then be connected. The leave operation is not showed in the pseudo-code.

[1] Recall that the connectivity of a graph is upper bounded by its minimal degree.

[2] Intuitively, the number $\lfloor \frac{|S|}{2} \rfloor$ was chosen for balancing the connectivities of u and v. Other choices could solve the problem and preserve the correctness of the solution as well. We intend to further explore this aspect in future work.

REPAIR() at node x
  if $\exists u \in neighbors : u$ is failed **then** send(PULL_NEIGHBORS($neighbors$) to $neighbors$;
upon receive PULL_NEIGHBORS() at node x from node y
  wait until $|neighbors| \geq \lfloor \frac{k}{2} \rfloor + 1$
  $NS \leftarrow \{u \in neighbors : u.is\_saturated = false\}$;
  $S \leftarrow neighbors - NS$;
  $X \leftarrow select(\lfloor \frac{|S|}{2} \rfloor, S)$;
  send PUSH_UPDATE($NS \cup X$) to y;
  send PUSH_UPDATE(new_node) to $NS \cup X$;
  $neighbors \leftarrow neighbors - X$;
upon receive PUSH_UPDATE(nodes) at node x from node y
  for each $v \in nodes$
    if $v.op = x.op$ **then**
      if $v \in neighbors$ **then return**;
      $neighbors \leftarrow neighbors \cup v$;
      if $|neighbors| = k$ **then**
        send saturated(true) to $neighbors$;
      else if $|neighbors| = k + 1$ **then**
        $neighbors \leftarrow neighbors - y$;

Figure 3: Pseudo-code of Repair operation and updates

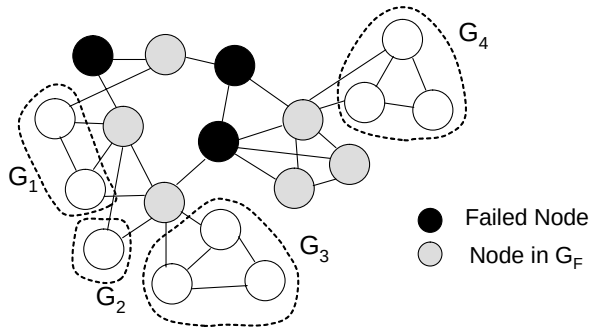


Figure 4: Nodes subdivision after a failure

5 Protocol Correctness

In this Section we prove that the built overlay can tolerate up to $\delta - 1$ concurrent failures before partitioning and is self-healing. We present a quick overview of the proofs in the following. We first analyze the effects of join operations, showing that they always lead to a topology satisfying $\delta-$connectivity (Lemmas 5.1 and 5.2). Then, we prove that starting from a $\delta-$connected topology, after $\delta - 1$ concurrent failures, the maintenance phase restores the $\delta-$connectivity of the overlay (Lemma 5.3).

5.1 Correctness

The correctness condition for our overlay is expressed in form of an invariant defining the connectivity within the system. Informally, the invariant states that if the overlay is δ-connected, any two nodes should be reachable by δ distinct paths.

Definition 5.1 (Connectivity Path) *Let S be an overlay defined by the neighbor relation. Let p and q be two nodes in S. p and q are connected iff $\exists p_0, p_1, \ldots p_k \in S$ such that: $p.neighbors = Q_0$, $p_0 \in Q_0 \wedge \forall i = 1, \ldots, k-1$, $p_i.neighbors = Q_{i+1}$, $p_{i+1} \in Q_{i+1} \wedge p_k.neighbors = Q, q \in Q$. The nodes $p_0, p_1, \ldots p_k$ are referred the connectivity path between p and q.*

Definition 5.2 (Distinct Connectivity Paths) *Let S be an overlay and let p and q be two nodes in S and let C_1 and C_2 two connectivity paths between p and q. C_1 and C_2 are distinct iff $C_1 \setminus C_2 \neq \emptyset$ or $C_2 \setminus C_1 \neq \emptyset$.*

Definition 5.3 (δ-connectivity invariant) *Let S be an overlay. S is δ-connected iff the following predicate is verified: $\forall p, q \in S$, exist δ distinct connectivity paths between p and q in S.*

5.2 Correctness Proofs

Given a set of nodes S, we denote by $\mathcal{G}_S$ the graph defined by the neighbors relations in the overlay computed by the protocol described in the previous section. We prove in the following the $\delta-$connectivity of $\mathcal{G}_S$, with $\delta = \lfloor \frac{k}{2} \rfloor + 1$.

Lemma 5.1 *Given a set of nodes S with n elements, $n \geq k$. If no leave operations or failures occur the minimum degree of $\mathcal{G}_S$ is δ.*

Proof. Suppose $n = k$. Then $\mathcal{G}_S$ is complete because any node gets all the existing nodes when it joins and can add them to its view. Suppose now that $n > k$. We assume by induction that the lemma is verified for $n = m - 1$ and consider $n = m$, with the last node joining G being node u. Node u joins an existing node v in G. From the inductive hypothesis, v has degree at least δ before u joins. Moreover, the sets S_v and NS_v of saturated and non-saturated neighbors of v are such that $\lfloor \frac{|S_v|}{2} \rfloor + |NS_v| \geq \frac{k}{2}$, otherwise v cannot add u. After u joins the group, the degree of v is $|S_v| + |NS_v| - \lfloor \frac{|S_v|}{2} \rfloor + 1 \geq \delta$ and the degree of u is $|NS_v| + \lfloor \frac{|S_v|}{2} \rfloor + 1 \geq \delta$. If u contacts a node w that cannot add it, u can always find v to join: in fact if w cannot add u only if $|S_w| > 0$, then w can forward u's request to one of its saturated nodes that will be able to add u, as it is easy to show.

Lemma 5.2 *Given a system S with n nodes, with $n \geq k$. If no leave operations or failures occur, $\mathcal{G}_S$ is δ-connected.*

Proof. Suppose $n = k$. Then $\mathcal{G}_S$ is complete, and the lemma is trivially true. Suppose now that $n > k$. Let us assume by induction that the lemma is verified for $n = m - 1$ and consider $n = m$, with the last node joining G being node u. A graph is $\delta-$connected if and only if it exists at least δ independent paths from any two nodes. From Lemma 5.1, node u has at least δ neighbors, each of which having at least δ distinct paths to any node from the inductive hypothesis. Considering one of the paths to each of u's neighbors plus the link to u, we obtain δ independent paths connecting any node to u.

Lemma 5.3 *Given a system S with n elements, $n \geq k$. If up to $\delta - 1$ locally concurrent failures occur in any neighborhood and no more failures occur during a repair phase, $\mathcal{G}_S$ is δ-connected at the end of the repair phase.*

Proof. Let F be the set of failed nodes of S, with $|F| \leq \delta - 1$. After removing F nodes from $\mathcal{G}_S$, the set of vertexes in $\mathcal{G}_S$ can be partitioned into the following set of vertexes as shown in Figure 4[3]: the set of nodes G_F that were neighbors to nodes in F and the sets G_i of nodes that were not all neighbors of nodes in F and such that for any couple p, q no nodes in G_p are connected to a node in G_q. From Lemma 5.2, $\mathcal{G}_S$ is still connected after the failures, then from definition all the G_q and G_p are connected only through a node in G_F. Nodes in G_F are those whose connectivity is hit by the failures, while each G_i is still $\delta-$connected. For any node v_F in G_F, two cases are possible:
i) v_F is connected to at least one node in G_i. v_F reacts to the failure by invoking a pull on its neighbor v_i in G_i. As it can be seen by comparing the code of the JOIN and PULL_UPDATE message handlers, the effect of a single pull request is equal to a join, provided the pull target has degree δ. From Lemma 5.1, this leads to a degree equals to δ for the joining node. Then after the pull v_F achieves δ links and merges into G_i.
ii) v_F is only connected to nodes in G_F. This means that it exists a connected subset G_{F_i} of G_F. As

[3] Please note that the depicted topology is purely indicative and is not intended to represent a realistic topology obtained by the protocol application.

any node in G_F may have less than δ node, nodes in G_{F_i} may not be able to reply to the pull. However, there will be at least one node v in G_{F_i} that is connected to a G_i, otherwise, G_{F_i} would be partitioned. As shown in case i), v is eventually merged to G_i, thus making its neighbors in G_{F_i} fall into case i) at the following step. Eventually, all the nodes blocked on the pull operation will release the block until G_{F_i} entirely merges into G_i.
We showed that any node in G_F is eventually integrated into one of the G_i sets. All the G_i being connected through nodes in G_F, this means that all the G_i merge into a single $\delta-$connected graph.

6 Conclusion

In this paper we have proposed a novel deterministic, δ-connected DHT-free P2P overlay. Our overlay offers strong connectivity guarantees despite system churn (i.e. frequent nodes join and leave or failures). The construction and the maintenance of our overlay is handled using only local information. Moreover, we have analytically proved the correctness of our solution. Additionally, we have identified the necessary and sufficient conditions in terms of leave and join concurrency in order to ensure without degradation the δ-connectivity property. Thus, our solution is suitable for application with strong load balancing and reliability constraints like for example video streaming.

7 Acknowledgments

We thank Emmanuelle Anceaume for insightfull discussions related to the importance of the connectivity issues in publish/subscribe applications.

References

[1] A.M. Kermarrec A. J. Ganesh and L. Moussoulie. Peer-to-peer membership management for gossip based-protocols. *IEEE Transactions on Computer Systems*, 21(4), 2003.

[2] A. Allavena, Alan Demers, and John E. Hopcroft. Correctness of a gossip based membership protocol. *PODC*, 2005.

[3] Paul C. Attie and Nancy A. Lynch. Dynamic input/output automata: A formal model for dynamic systems. In *CONCUR*, pages 137–151, 2001.

[4] Roberto Baldoni, Adnan Noor Mian, Sirio Scipioni, and Sara Tucci Piergiovanni. Churn resilience of peer-to-peer group membership: A performance analysis. In *IWDC*, pages 226–237, 2005.

[5] Miguel Castro, Manuel Costa, and Antony I. T. Rowstron. Performance and dependability of structured peer-to-peer overlays. In *DSN*, pages 9–18, 2004.

[6] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, Animesh Nandi, Antony I. T. Rowstron, and Atul Singh. Splitstream: High-bandwidth content distribution in cooperative environments. In *IPTPS*, pages 292–303, 2003.

[7] Yatin Chawathe, Sylvia Ratnasamy, Lee Breslau, Nick Lanham, and Scott Shenker. Making gnutella-like p2p systems scalable. In *SIGCOMM*, pages 407–418, 2003.

[8] Patrick Th. Eugster, Rachid Guerraoui, Sidath B. Handurukande, Petr Kouznetsov, and Anne-Marie Kermarrec. Lightweight probabilistic broadcast. In *DSN*, pages 443–452, 2001.

[9] Ayalvadi J. Ganesh, Anne-Marie Kermarrec, and Laurent Massoulié. Scamp: Peer-to-peer lightweight membership service for large-scale group communication. In *Networked Group Communication*, pages 44–55, 2001.

[10] M. Frans Kaashoek and David R. Karger. Koorde: A simple degree-optimal distributed hash table. In *IPTPS*, pages 98–107, 2003.

[11] Jon M. Kleinberg. The small-world phenomenon: an algorithm perspective. In *STOC*, pages 163–170, 2000.

[12] Dejan Kostic, Adolfo Rodriguez, Jeannie R. Albrecht, and Amin Vahdat. Bullet: high bandwidth data dissemination using an overlay mesh. In *SOSP*, pages 282–297, 2003.

[13] Dahlia Malkhi, Moni Naor, and David Ratajczak. Viceroy: a scalable and dynamic emulation of the butterfly. In *PODC*, pages 183–192, 2002.

[14] Roie Melamed and Idit Keidar. Araneola: A scalable reliable multicast system for dynamic environments. In *NCA*, pages 5–14, 2004.

[15] Vern Paxson. End-to-end internet packet dynamics. In *SIGCOMM*, pages 139–152, 1997.

[16] Sylvia Ratnasamy, Mark Handley, Richard Karp, and Scott Shenker. Application-level multicast using content-addressable networks. *LNCS*, 2233:14–34, 2001.

[17] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proc. of the 4th IFIP/ACMv Middleware Conference (Middleware '01)*, pages 329–350, 2001.

[18] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proc. of ACM SIGCOMM*, 2001.

[19] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Transactions on Networking*, 11(1), February 2003.

[20] Spyros Voulgaris, Daniela Gavidia, and Maarten van Steen. Cyclon: Inexpensive membership management for unstructured p2p overlays. *J. Network Syst. Manage.*, 13(2), 2005.